Gaussian Process Regression in Finance: From Dynamic Incentive Models to Portfolio Optimization

Simon Scheidegger[1]

[1]Department of Economics, University of Lausanne

Deep Learning for Economics and Finance
University of Geneva; March 25th - 27th, 2025

Outline of the Talk

1. Introduction and Motivation

2. Gaussian Process Regression: A Primer

3. Two Dynamic Models
    1. Machine Learning for Dynamic Incentive Problems (Renner and Scheidegger, 2018)
    2. A Comprehensive Machine Learning Framework for Dynamic Portfolio Choice With Transaction Costs (Gaegauf et al., 2023)

4. Key Takeaways and Conclusion

Dynamic Models in Finance and Economics

Many contemporary research questions are addressed via dynamic structural models, such as:

- Dynamic Incentive Problems
- Monetary Policy
- Climate Change

$\rightarrow$ **Rich formulated economic environments**:

- Models are often stochastic
- Models consist of many agents
- Nonlinear (e.g., due to financial frictions)
- Many Parameters (structural estimation)

Our Friend, the Bellman Equation

- **Dynamic Models** are often defined via the Bellman equation, characterizing the value function $V(s)$ for a state s.

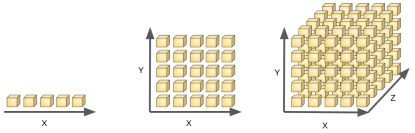
$$V(s) = \max_{a \in A(s)} \left\{ R(s,a) + \beta \int_S V(s')\, P(s'|s,a)\, ds' \right\}$$

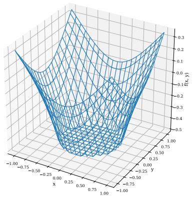
- $V(s)$: Value function, representing maximum expected reward from state s.
- a: Action taken in state s.
- $A(s)$: Set of allowable actions in s.
- $R(s,a)$: Immediate reward for action a in state s.
- β: Discount factor, $0 < \beta \leq 1$, weighting future rewards.
- $P(s'|s,a)$: Transition probability to state s' given s and a.
- S: State space, which can be d-dimensional with large d.

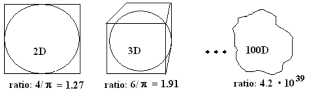
$\rightarrow$**This recursive equation is often solved with Value Function Iteration**.

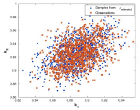
Numerical Roadblocks of Dynamic Models

1. Models suffer from the curse of dimensionality (many state variables,...).
2. Models suffer from non-linearities (financial frictions,...).
3. Have to approximate and interpolate high-dimensional functions on irregular-shaped geometries.
4. For high dimensions: ratio of Volume(Sphere)/Volume(Cube) $\rightarrow 0$.
5. **If projection methods/DP are used, solving optimization problems is expensive.**









$\rightarrow$ **One way out: Gaussian Processes** (Rasmussen and Williams, 2005).

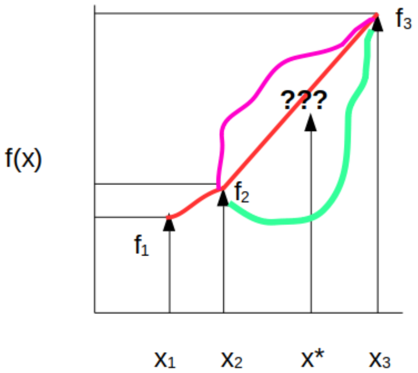
Gaussian Process Regression: A Primer

Function Approximation with Gaussian Processes

- Want to approximate Value- and Policy functions
  - in high dimensions
  - that are nonlinear
  - on irregular geometries
- Gaussian Process (GP) regression is a non-parametric method used to approximate functions based on observed data that can satisfy this.
- Given training data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, we aim to infer the underlying function $f(x)$.

$$y_i = f(x_i) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

Intuition for GPs



We assume that f's (heights) are Gaussian distributed with zero mean and some covariance matrix K:

$$\begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \right)$$

Note: f_1 and f_2 should be more correlated due to proximity (compared to f_1 and f_3). Example:

$$\sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{bmatrix} 1 & 0.7 & 0.2 \\ 0.7 & 1 & 0.6 \\ 0.2 & 0.6 & 1 \end{bmatrix} \right)$$

Covariance matrix based on a **kernel function**, e.g.,:

$$\kappa(x, x') = \sigma_f^2 \exp\left(-\frac{1}{2\ell^2}(x - x')^2 \right)$$

where:

- σ_f^2: Controls vertical variation.
- ℓ: Controls horizontal length scale.

Defining the Gaussian Process Prior

- A GP is defined by a mean function $m(x)$ and a covariance function $k(x, x')$:

$$f(x) \sim \mathcal{GP}(m(x), k(x, x'))$$

- Commonly, we set $m(x) = 0$ to simplify computation, focusing on the covariance function.
- For any set of inputs $\{x_1, \ldots, x_N\}$, the function values $\{f(x_1), \ldots, f(x_N)\}$ follow a multivariate Gaussian distribution:

$$\mathbf{f} \sim \mathcal{N}(0, K)$$

where $K_{ij} = k(x_i, x_j)$.

Posterior Inference for Gaussian Process Regression

- Given training data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, we want to predict $f(x^*)$ at a new point x^*.
- The joint distribution of observed values $\mathbf{y}$ and the prediction $f(x^*)$ is:

$$\begin{bmatrix} \mathbf{y} \\ f(x^*) \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K + \sigma^2 I & k_* \\ k_*^\top & k(x^*, x^*) \end{bmatrix}\right)$$

  where $k_* = [k(x^*, x_1), \ldots, k(x^*, x_N)]$.
- The conditional distribution of $f(x^*) \mid \mathbf{y}$ is:

$$f(x^*) \mid \mathbf{y} \sim \mathcal{N}(\mu_{f(x^*)}, \sigma^2_{f(x^*)}),$$

  where the predictive mean is given by:

$$\mu_{f(x^*)} = k_*^\top (K + \sigma^2 I)^{-1} \mathbf{y}$$

  and the predictive variance reads as:

$$\sigma^2_{f(x^*)} = k(x^*, x^*) - k_*^\top (K + \sigma^2 I)^{-1} k_*$$

Predictive Mean and Variance

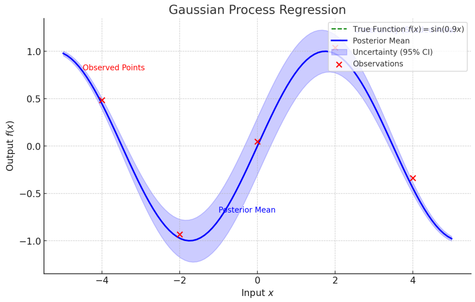
- To interpolate the function at any new point x^*, we use the **predictive mean** of the GP as our best estimate:

$$\text{Predictive Mean:} \quad \mathbb{E}[f(x^*)] = k_*^\top (K + \sigma^2 I)^{-1} \mathbf{y}.$$

- The **predictive variance** quantifies the uncertainty around this estimate, providing confidence intervals:

$$\text{Predictive Variance:} \quad \text{Var}(f(x^*)) = k(x^*, x^*) - k_*^\top (K + \sigma^2 I)^{-1} k_*.$$

- **Together, these expressions allow us to both estimate the function's value at any x^* and understand the associated uncertainty**.



Choosing the Prior Kernel Function

- The kernel function $k(x, x')$ determines the properties of the GP, such as smoothness and periodicity.
- Common choices include:
  1. **Squared Exponential (SE) Kernel**:
     $$k(x, x') = \sigma^2 \exp\left(-\frac{(x - x')^2}{2\ell^2}\right)$$
     where ℓ controls smoothness and σ controls output variance.
  2. **Matérn Kernel** (provides more flexibility in smoothness):
     $$k(x, x') = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}|x - x'|}{\ell}\right)^{\nu} K_{\nu}\left(\frac{\sqrt{2\nu}|x - x'|}{\ell}\right)$$
     where ν determines the differentiability.
  3. **Periodic Kernel**: captures repeating patterns, useful in seasonal data.
  4. Kernel Cookbook: https://www.cs.toronto.edu/~duvenaud/cookbook/

Hyperparameter Optimization in GPR

- **Goal**: Optimize the hyperparameters of the kernel function to improve the fit of the GP model.
- **Process**: Hyperparameters are tuned by maximizing the marginal likelihood of the observed data:

$$\log p(\mathbf{y}|\mathbf{X}, \theta) = -\frac{1}{2}\mathbf{y}^\top (K_\theta + \sigma^2 I)^{-1}\mathbf{y} - \frac{1}{2}\log\det(K_\theta + \sigma^2 I) - \frac{n}{2}\log(2\pi)$$

  where θ represents the hyperparameters of the kernel K_θ.
- **Common Optimizers**:
  - **Gradient Descent**: Uses the gradient of the marginal likelihood to adjust hyperparameters iteratively.
  - **L-BFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno)**: A popular quasi-Newton method efficient for high-dimensional optimization.
  - **Adam (Adaptive Moment Estimation)**: An adaptive learning rate optimizer often used in machine learning, balancing speed and stability.

Bayesian Active Learning with Gaussian Processes

$\rightarrow$ **Uniform sampling in a high-dimensional space to populate the training set might be inefficient**.

- **Goal**: Given a fixed computational budget, efficiently approximate functions using GPs by actively selecting data points that improve the model's accuracy.
- **Core Idea**: Rather than uniformly sampling, BAL strategically adds observations in areas where they contribute most to the model's performance (e.g., MacKay, 1992, Chaloner and Verdinelli, 1995, Krause et al., 2008, Deisenroth et al., 2009, and Makarova et al., 2022).
- **Application**: Particularly useful in settings like value function iteration, where acquiring **new data points is computationally costly** (each training sample consists of solving a constrained optimization problem).

How Bayesian Active Learning Works

1. **Model Update**: Fit a GP to current data $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$ to obtain a posterior mean and uncertainty for each point.
2. **Acquisition Function**: Evaluate candidate points by their expected contribution to the model, often using a score like:

$$\alpha(\tilde{x}_j; \mathcal{D}) = \sigma_m \tilde{\mu}(\tilde{x}_j) + \frac{\sigma_v}{2} \log\left(\tilde{\sigma}(\tilde{x}_j)\right)$$

3. **Data Selection**: Select the candidate with the highest score, add it to the dataset, and update the model.[1]

[1]For effective use of the acquisition function, one needs to **randomly sample a set of candidate points** (e.g., 100), rank them by their acquisition scores, and add the highest-ranked point(s) to the training set.

BAL Process Illustrated

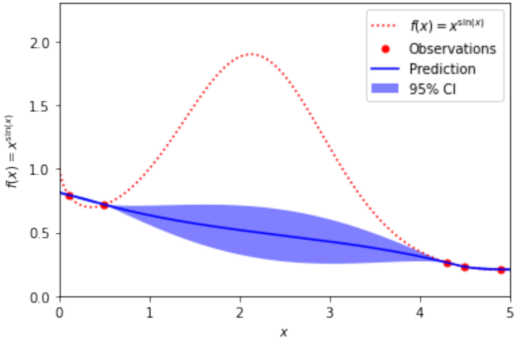


Figure: Phased visualization of Bayesian Active Learning.

BAL Process Illustrated

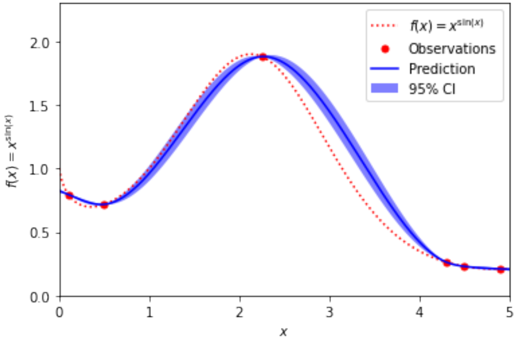


Figure: Phased visualization of Bayesian Active Learning.

BAL Process Illustrated

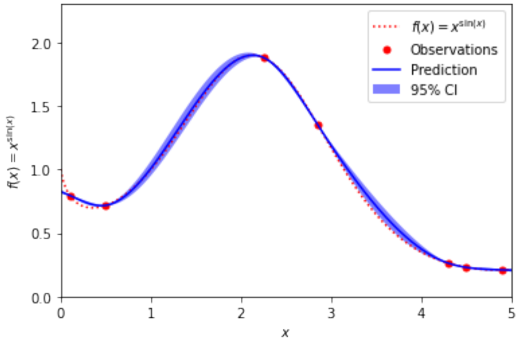


Figure: Phased visualization of Bayesian Active Learning.

Summary of Bayesian Active Learning Process

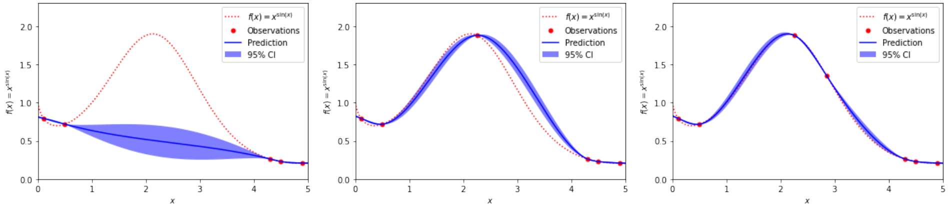
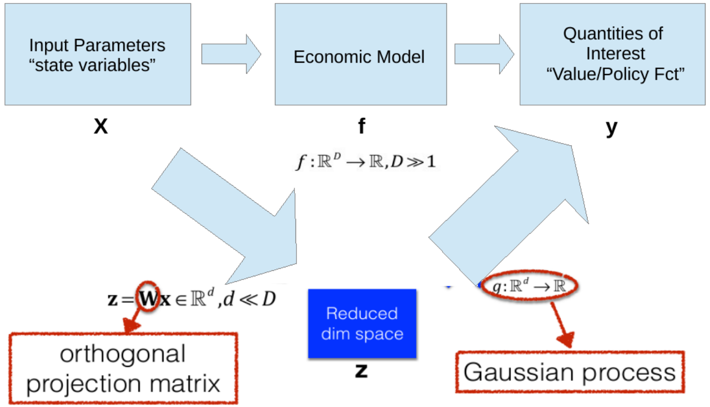


Figure: Phased Bayesian Active Learning: Initial data, refined with incremental observations via BAL.

Active Subspace-Based Dimension Reduction

- For dimensions larger than about 10, GPs drastically lose performance (Euclidean distance becomes uninformative in high-dim spaces).
- Need a formal way for reducing dimensionality.
- See Constantine et al. (2014) in general, Scheidegger and Bilionis (2019) for DP.



Input Parameters "state variables"
Economic Model
Quantities of Interest "Value/Policy Fct"
x
f
y
$f:\mathbb{R}^D \rightarrow \mathbb{R}, D \gg 1$
$\mathbf{z} = \mathbf{W}\mathbf{x} \in \mathbb{R}^d, d \ll D$
Reduced dim space
z
$g:\mathbb{R}^d \rightarrow \mathbb{R}$
orthogonal projection matrix
Gaussian process

Figure: Intuition behind Active Subspaces.

Active Subspaces – intuitive example

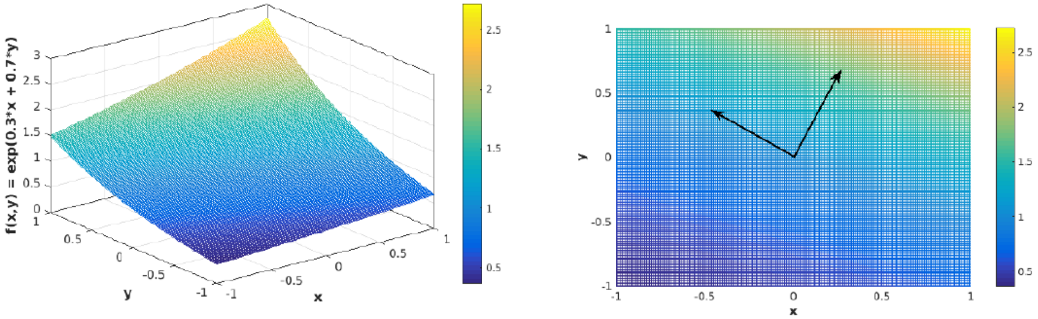


Figure: Function varies most along [0.3, 0.7], and is constant in the orthogonal direction.

Discover Active Subspaces

See, e.g., Constantine (2015), with references therein

Step 1: Find W

$$\mathbf{C} = \mathbb{E}\left[\nabla_x f(x) \nabla_x f(x)^T\right] \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_x f(x^{(i)}) \nabla_x f(x^{(i)})^T$$

$$\mathbf{C} = \mathbf{W}\Lambda\mathbf{W}^T$$

- **"Mean-square directional derivative"**
- Note: derivative-free methods also exist for constructing **W**.

Discover Active Subspaces (cont.)

Step 2: Partition the Eigendecomposition

$$\Lambda = \begin{bmatrix} \Lambda_1 & \\ & \Lambda_2 \end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix} \mathbf{W}_1 & \mathbf{W}_2 \end{bmatrix}, \quad \mathbf{W}_1 \in \mathbb{R}^{m \times n}$$

- Compute Eigenvalues, order them.
- Look for "gaps."

Step 3: Create a Rotated Coordinate System

$$\mathbf{x} = \mathbf{W}\mathbf{W}^T\mathbf{x} = \mathbf{W}_1\mathbf{W}_1^T\mathbf{x} + \mathbf{W}_2\mathbf{W}_2^T\mathbf{x} = \mathbf{W}_1\mathbf{q} + \cancel{\mathbf{W}_2\mathbf{v}}$$

Dimension Reduction: An Example

$$f : [-1,1]^{10} \to \mathbb{R}$$

$$f(x_1, \dots, x_{10}) = \exp(0.01x_1 + \boxed{0.7x_2} + 0.02x_3 + 0.03x_4 + 0.04x_5 + 0.05x_6 + 0.06x_7 + 0.08x_8 + 0.09x_9 + 0.1x_{10}$$

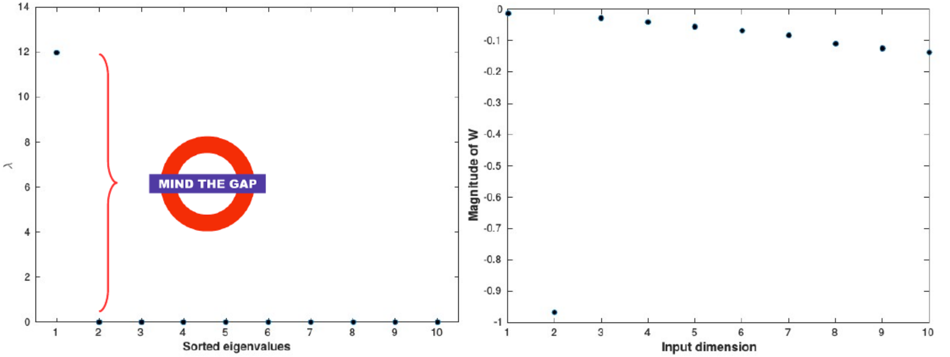


Figure: Active Subspaces in Action.

Gaussian Processes in Perspective

- GPs, together with BAL (and active subspaces), provide a tool to approximate high-dimensional, nonlinear functions on irregular geometries.
- This is particularly useful when training data is costly to acquire (each data point corresponds to solving a constrained optimization problem).
- This combination provides a generic tool for value function iteration.

Table: GPs in perspective with other function approximators.

Properties	**Gaussian Process Regression**	**Neural Networks**	**Adaptive Sparse Grids**	**Chebyshev Polynomials**
Handles High Dimensionality	✓	✓	✓	×
Nonlinear Function Approximation	✓	✓	✓	✓
Probabilistic Predictions	✓	×	×	×
Scalability with Data Size	×	✓	✓	×
Requires Grid Construction	×	×	✓	×
Adaptive Refinement	✓	×	✓	×
Interpretability	✓	×	✓	✓
Parallelization Potential	✓	✓	✓	✓
Hyperparameter Tuning Needed	Moderate	High	Low	Low
Uncertainty Quantification	✓	×	Limited	×

Two Dynamic Models

1. Dynamic Incentive Problems

Dynamic incentive problems:

- They occur whenever **two parties with repeated interaction under asymmetric information form a contract**.
- Applications: insurance contracts, optimal taxation, manager remuneration, . . .
- e.g., a **risk averse agent** buys insurance from a **risk neutral planner** against income shocks (which are hidden to the planner).

i) **adverse selection**: there is *hidden information* (to one party in the contract).

ii) **moral hazard**: there is a *hidden action*.

Example below: dynamic adverse selection problems with persistent shocks

→ Solving such models numerically is a formidable task.

→ Models are often formulated in a *stylized* fashion to remain computationally **tractable**.

A simple dynamic optimal insurance model

see e.g. *Fernandes and Phelan (2000); Golosov et al. (2016)*

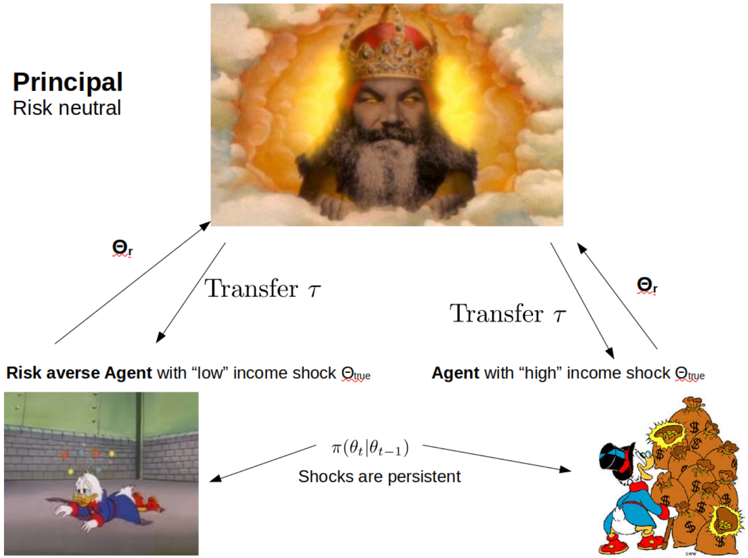
A prototypical insurance model:

- Infinite-horizon, discrete time model.
- **Risk-averse agent**: faces *privately-observed* income shocks Θ.
- Income shocks are **persistent**.
- **Risk neutral principal**: wants to provide optimal *incentive-compatible insurance* against income shocks.
- **Agent**: reports his income shock to the principal.
- **Principal**: *transfers consumption* or *charges a fee* to the agent, dependent on report.

Optimal solution to the problem:

$\rightarrow$ **A transfer scheme that maximizes the principal's utility while delivering a predetermined lifetime utility to the agent.**

A Prototypical Dynamic Incentive Model


Principal
Risk neutral
Θ_r
Transfer τ
Transfer τ
Θ_r
Risk averse Agent with "low" income shock Θ_{true}
Agent with "high" income shock Θ_{true}
$\pi(\theta_t|\theta_{t-1})$
Shocks are persistent

An auxiliary problem formulation

Formal issues:

- The observed reports are fully history dependent.
- The principal needs to be prepared for infinitely many reporting strategies.

$\rightarrow$ **there is no obvious recursive formulation for the model.**

To make the problem formally tractable (following existing literature):

$\rightarrow$ Apply the *revelation principle* and *one-shot deviation* principle.

$\rightarrow$ Introduce an auxiliary problem with promised expected utilities as state variables.

$\rightarrow$ To discipline the agent, we need promise keeping and threat keeping as states (lying always needs to be sub-optimal).

$\rightarrow$ For every type of shock in the model, we need an additional state (if there are N types, there are $N-1$ possibilities to lie).

$\rightarrow$ The continuous state space of the auxiliary problem is N-dimensional.

Adverse selection: why difficult to solve?

Addressing **dynamic adverse selection models with persistent shocks** is a formidable task, since **two major computational bottlenecks** create difficulties in the solution process.

1) The **feasible sets** of utility promises are not known in advance and have to be determined numerically.

2) For solving dynamic adverse selection problems with value function iteration, we need to repeatedly approximate and evaluate high-dimensional functions at arbitrary coordinates within the feasible sets.

$\rightarrow$ Use-case for Gaussian Process Dynamic Programming.

Our solution method

- Generic penalty reformulation of dynamic adverse selection problems.

$$\bar{K}\left(\hat{v}(\cdot), \theta_{-}\right) = \max_{c(\cdot), w(\cdot|\cdot), \xi_j} \sum_{\theta \in \Theta} \pi\left(\theta|\theta_{-}\right)\left[v(c(\theta), \theta_{-}) + \beta \bar{K}(w(\theta|\cdot), \theta)\right] - M \sum_{j=1}^{m} \xi_j^2$$

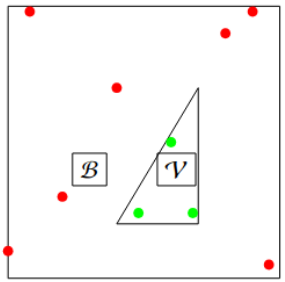
$$\hat{v}\left(\theta_{(j)}\right) + \xi_j = \sum_{\theta \in \Theta} \pi\left(\theta|\theta_{(j)}\right)[U(c(\theta), \theta) + \beta w(\theta|\theta)], \quad \forall j \in \{1, \ldots, m\}$$

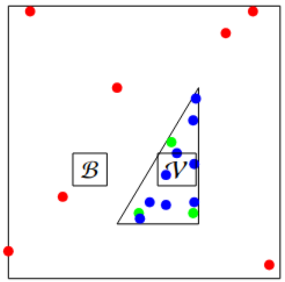
$$U(c(\theta), \theta) + \beta w(\theta|\theta) \geq U(c(\hat{\theta}), \theta) + \beta w(\hat{\theta}|\theta), \quad \forall \theta, \hat{\theta} \in \Theta$$

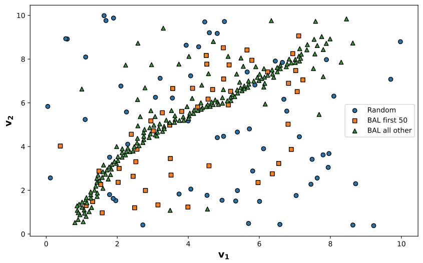
$$c(\theta) \in C, w(\theta|\cdot) \in \mathcal{B} = \prod_{\tilde{\theta} \in \Theta}[U(\underline{c}, \tilde{\theta})/(1-\beta), U(\bar{c}, \tilde{\theta})/(1-\beta)], \quad \forall \theta \in \Theta$$

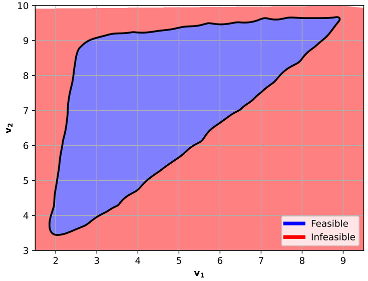
- This reformulation has the advantage that computing solutions to the reformulated problem reduces to solving an ordinary dynamic programming problem via value function iteration.
- No need to explicitly characterize the feasible set.
- So far: Set-valued DP (Abreu et al., 1986); geometric constructions, don't scale.
- Use GPs to approximate value and policy functions.

Intuition of method (works for at least ten types)





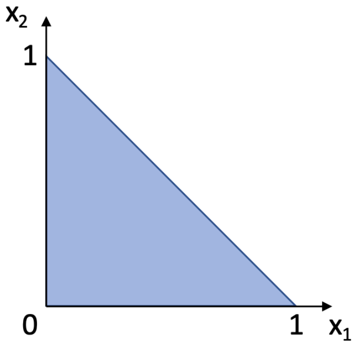


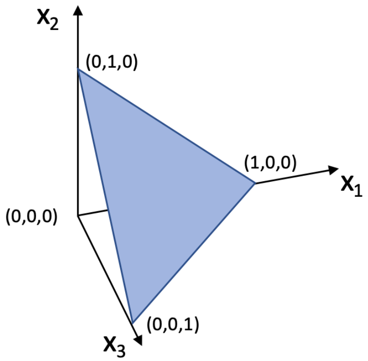


2. Dynamic Portfolio Choice with Transaction Costs

- Optimization of intertemporal utility subject to dynamic budget constraint.
- Solution characterized by means of associated Bellman equation.
- Solvable in closed-form under stringent assumptions about, e.g., utility, return dynamics, market completeness, and cost structure.
- Existing numerical solution methods are limited by the curse of dimensionality.
- Existing numerical solution methods hardly applicable with more than 3-4 risky assets.
- Intrinsically "distinct" optimal portfolio behavior over an irregularly-shaped domain.
- Not everywhere differentiable optimization problem with varying smoothness and nonlinearity properties over admissible domain.

Some Issues: Curse of Dimensionality, and Irregularly-Shaped Domain





- Budget constraint depends on D-dimensional vector of risky assets' wealth shares.
- Irregularly-shaped domain with, e.g., no-short selling/borrowing constraints.
- A challenge for grid-based function approximation methods.

Objective

A comprehensive machine learning solution framework for dynamic portfolio choice with proportional transaction costs (Gaegauf et al., 2023):

- Scalable.
- Accomodating nonlinearities, nondifferentiabilities and irregularly-shaped domains.
- "Tunable-smooth" over different subregions of the domain.
- Allowing a quantification of the uncertainty of computed solutions.
- Extendable, e.g., to general equilibrium and state-dependent opportunity sets.

$\rightarrow$ Use-case for GPs.

Recursive form and Bellman equation

1. For some economically motivated terminal utility function $U(\cdot,\cdot)$, let:

$$V_T(W_T, \boldsymbol{x}_T) := U(W_T, \boldsymbol{x}_T) \tag{1}$$

2. Define recursively, for any $t = T-1, \dots, 0$:

$$V_t(W_t, \boldsymbol{x}_t) = \max_{(C_t, \boldsymbol{\delta}_t) \in \mathcal{D}(W_t, \boldsymbol{x}_t)} \left\{ u(t, C_t) + \mathbb{E}[V(W_{t+1}, \boldsymbol{x}_{t+1}) | W_t, \boldsymbol{x}_t] \right\} , \tag{2}$$

   where

$$(W_{t+1}, \boldsymbol{x}_{t+1}) = BC(W_t, \boldsymbol{x}_t, C_t, \boldsymbol{\delta}_t; R_f, \boldsymbol{R}_{t+1})$$

3. Obtain desired value function and optimal controls from last iteration at $t = 0$.

Key Issues for Computational Solution Framework

1. "Exact" computation of $V_t(\cdot,\cdot)$ over entire domain $\mathcal{D}_t$ is infeasible/inappropriate.

2. Need approximation of $V_t(\cdot,\cdot)$ from suitable set of "exact" computations $\{V_t(W_i, \boldsymbol{x}_i)\}_{i=1}^N$:

   - Error in approximation of $V_t(\cdot,\cdot)$ needs to be quantifiable.
   - Accuracy to bound error propagation in global solution as, e.g., D or T grows.
   - Capture varying degrees of nonlinearity/nondifferentiability over domain $\mathcal{D}_t$.
   - Efficiency with "exact" computations performed only within irregularly-shaped domain $\mathcal{D}_t$.

3. Grid-free probabilistic approximation methods are preferable.

Comprehensive Machine Learning Framework

- Based on recursive grid-free probabilistic approximation of value function $V_t(\cdot,\cdot)$.
- Two building blocks:

1. Accurate description of unknown NTR:

$$\boldsymbol{\Omega}_t := \{(W_t, \boldsymbol{x}_t) \in \mathcal{D}_t : \boldsymbol{\delta}_t^{opt} = \boldsymbol{0}\}$$

   by means of suitable approximate NTR $\hat{\boldsymbol{\Omega}}_t$.

2. Accurate approximation of $V_t(\cdot,\cdot)|_{\hat{\boldsymbol{\Omega}}_t}$ and $V_t(\cdot,\cdot)|_{\mathcal{D}_t \setminus \hat{\boldsymbol{\Omega}}_t}$ using two different associated Gaussian Process Regressions (GPRs).

Value Function Approximation with two GPRs

- Account for intrinsically distinct value function properties inside/outside NTR:

$$v_{1t} := v_t \mathbf{1}_{\hat{\mathbf{\Omega}}_t} \ ; \ v_{2t} := v_t \mathbf{1}_{\mathcal{D}_t \setminus \hat{\mathbf{\Omega}}_t}$$

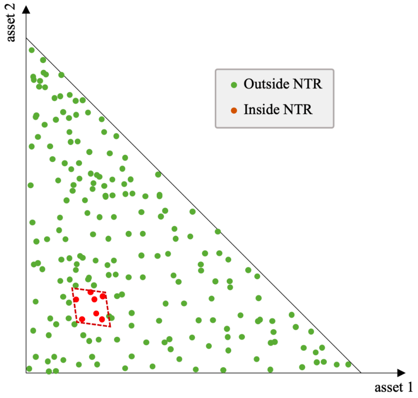
- "Disaggregated" Bellman equation: For any $t = T-1, \dots, 0$:

$$v_{1t}(\mathbf{x}_t) + v_{2t}(\mathbf{x}_t) = \max_{(c_t, \boldsymbol{\delta}_t) \in \mathcal{D}(\mathbf{x}_t)} \left\{ u(c_t) + \beta \mathbb{E}\left[\pi_{t+1}^{1-\gamma} \left(v_{1t+1}(\mathbf{x}_{t+1}) + v_{2t+1}(\mathbf{x}_{t+1}) \right) | \mathbf{x}_t \right] \right\}$$

- Two distinct GPRs for approximating the value function inside/outside NTR:

$$\mathcal{GP}_{1t}(\mathbf{x}_t) + \mathcal{GP}_{2t}(\mathbf{x}_t) = \max_{(c_t, \boldsymbol{\delta}_t) \in \mathcal{D}(\mathbf{x}_t)} \left\{ u(c_t) + \beta \mathbb{E}\left[\pi_{t+1}^{1-\gamma} \left(\mathcal{GP}_{1t+1}(\mathbf{x}_{t+1}) + \mathcal{GP}_{2t+1}(\mathbf{x}_{t+1}) \right) \right.\right.$$

Putting Things Together



- Randomly generate training samples of points inside/outside NTR.
- Train a distinct GPR on each subsample and predict remaining value function and policy values inside/outside NTR.

Algorithm

```
Input: Terminal value function $v_T$, time horizon $T$, sampling size $N$
Output: Surrogate value functions $\{\mathcal{V}_{t-1}\}_{t=1}^{T}$ and approximate NTRs $\{\hat{\mathbf{\Omega}}_{t-1}\}_{t=1}^{T}$

1. Set $\mathcal{V}_T = v_T$
2. For $t = T$ to $1$ do:
   Compute approximate NTR $\hat{\mathbf{\Omega}}_{t-1}$ using $\mathcal{V}_t$ as next period's value function and
   sample $N$ points $\mathbf{X}_{t-1} = \{\mathbf{x}_{t-1i}\}_{i=1}^{N} \in \mathcal{D}_{t-1}^{N}$
        For $i = 1$ to $N$ do:
        Obtain value function and policy values $(\hat{v}_{t-1i}, \hat{c}_{t-1i}, \hat{\boldsymbol{\delta}}_{(t-1)i})$ for $\mathbf{x}_{t-1i}$ by solving the
        Bellman equation using $\mathcal{V}_t$ as next period's value function.
        end
   Fill training sets $\hat{\mathcal{D}}_{1t}$, $\hat{\mathcal{D}}_{2t}$ based on whether $\mathbf{x}_{t-1i} \in \hat{\mathbf{\Omega}}_{t-1}$ or $\mathbf{x}_{t-1i} \notin \hat{\mathbf{\Omega}}_{t-1}$ for
   some $i$
   Learn from $\hat{\mathcal{D}}_{1t}$, $\hat{\mathcal{D}}_{2t}$ two GPR surrogates $\mathcal{V}_{1(t-1)}, \mathcal{V}_{2(t-1)}$ of $v_{1(t-1)}$, $v_{2(t-1)}$, to
   finally obtain a surrogate $\mathcal{V}_{t-1} = \mathcal{V}_{1(t-1)} + \mathcal{V}_{2(t-1)}$ of $v_{t-1}$.
   end
```

Key Takeaways and Conclusion

Main Points to Remember

Common Methodological Takeaways:

- Combining **dynamic programming** with **GPs** for efficient approximation of value and policy functions on irregular geometries is a powerful, scalable tool for solving dynamic models when data is expensive to acquire.
- **BAL** can boost performance by strategically focusing computational resources.
- **Active Subspaces** can help to reduce the dimensionality of problems effectively.
- **Grid-free approach** provides flexibilty.
- Allows the **characterization of irregular state spaces**, such as no-trade regions or feasible sets in high-dimensional settings.

Examples Highlighting the Method's Versatility:

- **Dynamic Incentive Problems (Renner and Scheidegger, 2018):** Presents a solution framework for dynamic adverse selection models with **persistent types**, offering, e.g., insights into adverse selection effects in insurance markets.
- **Dynamic Portfolio Choice (Gaegauf et al., 2023):** Applies the framework to **multi-asset portfolio optimization with transaction costs**, demonstrating that a broader asset space can mitigate liquidity constraints due to transaction costs.

Thank you! Questions?

References I

Abreu, D., Pearce, D., and Stacchetti, E. (1986). Optimal cartel equilibria with imperfect monitoring. *Journal of Economic Theory*, 39(1):251 – 269.

Chaloner, K. and Verdinelli, I. (1995). Bayesian experimental design: A review. *Statist. Sci.*, 10(3):273–304.

Constantine, P. G., Dow, E., and Wang, Q. Q. (2014). Active subspace methods in theory and practice: Applications to kriging surfaces (vol 36, pg a1500, 2014). *Siam Journal on Scientific Computing*, 36(6):A3030–A3031.

Deisenroth, M. P., Rasmussen, C. E., and Peters, J. (2009). Gaussian process dynamic programming. *Neurocomputing*, 72(7):1508–1524.

Fernandes, A. and Phelan, C. (2000). A recursive formulation for repeated agency with history dependence. *Journal of Economic Theory*, 91(2):223 – 247.

Gaegauf, L., Scheidegger, S., and Trojani, F. (2023). A comprehensive machine learning framework for dynamic portfolio choice with transaction costs. *Available at SSRN 4543794*.

References II

Golosov, M., Tsyvinski, A., and Werquin, N. (2016). Recursive contracts and endogenously incomplete markets. *Handbook of Macroeconomics*, 2:725–841.

Krause, A., Singh, A., and Guestrin, C. (2008). Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *J. Mach. Learn. Res.*, 9:235–284.

MacKay, D. J. C. (1992). Information-based objective functions for active data selection. *Neural Computation*, 4(4):590–604.

Makarova, A., Shen, H., Perrone, V., Klein, A., Faddoul, J. B., Krause, A., Seeger, M., and Archambeau, C. (2022). Automatic termination for hyperparameter optimization. Technical report.

Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Renner, P. and Scheidegger, S. (2018). Machine learning for dynamic incentive problems. Working paper. Available at SSRN: http://dx.doi.org/10.2139/ssrn.3282487.

References III

Scheidegger, S. and Bilionis, I. (2019). Machine learning for high-dimensional dynamic stochastic economies. *Journal of Computational Science*, 33:68 – 82.